HRTS: a Hierarchical Reactive Time Synchronization protocol for wireless sensor networks

Saeid Pourroostaei Ardakani, Julian Padget, and Marina De Vos

Computer Science Department,
University of Bath, UK
spa23,jap,mdv@cs.bath.ac.uk

Abstract. Time synchronization plays an important role in the performance of wireless sensor networks. It can enhance the throughput and the lifetime of the network by improving the energy-efficiency, the freshness of collected data and reducing the network traffic and message conflicts. Due to the constraints on sensor nodes' energy resources and the vulnerability of the distributed infrastructure of wireless network, an efficient, scalable and accurate time synchronization protocol is desirable. This paper presents an accurate and efficient reactive protocol, named HRTS, that synchronizes the sensor nodes' clock based on the node's demand. It minimizes the synchronization region dynamically to the set of nodes which request synchronization. HRTS improves the accuracy of the synchronization procedure by measuring time parameters accurately and removing delays accordingly. Compared with the conventional time synchronization protocols like PCTS and TPSN, HRTS also reduces energy consumption by decreasing the traffic overheads.

Keywords: wireless sensor networks, time synchronization protocol, end-to-end delay, on-demand synchronization.

1 Introduction

Wireless Sensor Networks (WSNs) are typically comprised of a large number of sensor nodes which are distributed in the sensing area and connected either randomly or hierarchically [1]. There is no specific distribution topology for random distributed WSN, whereas sensor nodes might be organized in a number of separate groups, called "clusters" in an hierarchical WSN. Sensor nodes typically have three main capabilities: sensing, processing, and communication [2]. The nodes usually measure/sense the environmental data and then transmit through either single or multi-hop paths to the end consumer's access point which is called the *Sink*. WSN is utilized for a set of diverse applications like environmental monitoring, education, surveillance, health and micro-surgery [3]. For example, in the case of environmental monitoring, this technology might be used to detect the air pollution, forest fire, level of humidity and natural disease.

Hardware imposed restrictions like the amount energy, memory, the communication radio range and processing power [4] need to be taken into account. Studies [5] show that both the computational and communicational tasks may consume the sensor node's power. Hence, decreasing the computational as well as communicational overheads to the greatest possible extent is the main goal of the research in WSNs.

Time synchronization has the ability to improve the network's energy efficiency by synchronizing sensor nodes to sense, process and communicate at certain intervals [6]. Using this technique, sensor nodes would be able to wake up at a specific time to perform processing and communication tasks and after that go to sleep to save the network energy. Owing to this, the nodes would consume the energy just over the specific period which they are awake to perform either the computation or communication tasks. On the other hand, the nodes might consume energy continuously to stay available over long period for performing the tasks when the network is not synchronized. Time sycnhronization also may improve the quality as well as freshness of data by collecting and monitoring sensor data at specific intervals which the consumer is interested in[7]. Owing to these issues, time synchronization can be considered as a common requirement for most WSNs applications like data aggregation and mobile object tracking to collect the realtime data.

The time on a sensor node is measured as a function of an oscillator and counter. The angular frequency of the oscillator increases the counter and so the value shows the time locally at each node [8]. The accuracy of angular frequency is very important for time accuracy. Any expected or unexpected changes and delays can lead to divergence between sensors' clocks [9, 10]. As Prakash and Kendall mention [6], there are three main reasons for the nodes to show different times: (1) the start time of nodes might be different (set-up error), (2) environmental parameters like vibration, temperature, pressure, and battery voltage change the angular frequency of the oscillator (drift error), (3) different operating frequency of quartz crystals of the nodes' clock might change the time (skew error).

In addition to these errors, communication delays can cause time uncertainties. As Miklos et al mention [11], there are nine time factors that may cause communication delays during synchronization procedure:

1 - Send time: time for assembling and transferring messages to the MAC layer on the transmitter side. It is nondeterministic (hundreds of milliseconds) and depends on the node's operating system.

2 - Access time: time to access the transmitting channels. It is deterministic and due to the network traffic can vary from milliseconds to seconds.

3 - Transmission time: time the transmitter spends to transmit the messages. It depends on the length of message and is the order of a few milliseconds.

4 - Propagation time: sensor nodes spend propagation time to transmit message from the transmitter channel of sender side to the receiver channel of other side. This time depends on the distance between the nodes.

5, 6 - Reception and receive time: respectively, the time of receiving messages by the MAC layer on the receiver side and passing the message to its application layer.
7 - Interrupt handling time: this is the response delay of sensor nodes' microcontroller to an incoming interrupt.
8, 9 - Encoding and decoding time: deterministic time spent to encode or decode a part or even the whole of the message.

This paper proposes a hierarchical reactive three-phase time synchronization protocol that takes into account the time parameters and delays. During the first phase, the protocol measures the time parameters locally at the sensor nodes. In the second-phase, it reduces the role of initial time on the sensor nodes' clocks to remove the set-up errors. Finally, it reduces the skew errors throughout the network by updating sensor nodes' local time reactively on demand.

Our approach is able to improve the throughput of synchronization by decreasing the energy consumption and synchronization errors. HRTS also may increase the network lifetime by reducing the number of transmissions. It means that, the synchronization transmissions would be limited within the synchronization regions which are created by the nodes (instead of the whole network) that ask for synchronization. In addition, decreasing the traffic overhead of synchronization messages may reduce the message conflicts in the network. So, a greater number of synchronization messages are delivered to the nodes correctly and the number of unsynchronized nodes is decreased.

The rest of paper is organized as follows: Section 2 describes some of the principal WSN time synchronization protocols. Section 3 presents HRTS and Section 4 discusses its performance evaluation. The last section concludes the paper and presents some plans for future work.

2 Related Work

There are a number of time synchronization protocols for WSN [6]. This section summurizes Reference Broadcast Synchronization (RBS) [12], Timing-synch Protocol for Sensor Network (TPSN) [10], Scalable Lightweight Time synchronization Protocol (SLTP) for wireless sensor network [13] and Passive Cluster based Clock Synchronization (PCTS) in sensor networks [14] as the most relevant protocols to HRTS in the term of hierarchical structure of synchronization.

Reference Broadcast Synchronization (RBS) is proposed by Elson et al [12]. It uses a sender-to-receiver scheme to synchronize the network nodes. In this protocol, the reference node synchronizes the network by broadcasting the time reference beacons. When a node receives the message, it records the local time and then exchanges the received time with its neighbours. The nodes estimate their clock offset and skew with the neighbours using linear regression methods. The main drawbacks of RBS are: (1) the protocol does not consider some of the time parameters/delays like access time in calculating the synchronized time. It does not consider network traffic to measure the access time which might increase skew errors especially when the network works over an extended period

of time, (2) RBS cannot be considered a scalable protocol for WSN, as sensor nodes would consume a great deal of energy to exchange the synchronization messages during the synchronization procedure, (3) readjusting the nodes' clock in RBS (updating the reference time) has high communication cost and also might increase time errors. Frequent updates of the reference time may increase the probability of message failures and skew errors due to messages conflicts at the reception time.

The Timing-synch Protocol for Sensor Networks (TPSN) is another protocol based on sender-to-receiver scheme [10]. TPSN consists of two main phases: level discovery and synchronization. During the first phase, a tree-based infrastructure is created by assigning level tags to the nodes hierarchically. In this phase, the node willing to become a reference node marks itself as level 0 and starts to broadcast the level-discovery packet throughout the network to create the hierarchical synchronization infrastructure. Each receiver increases the level value by one and then re-sends the level-discovery message to its neighbours. Then, this process is repeated until hierarchical levels are assigned to all nodes. In the second phase, the source node starts to synchronize its single-hop neighbours using a two-way message exchange. The synchronized nodes continue this procedure repeatedly until all the nodes are synchronized. TPSN has the potential to synchronize a large network in a multi-hop style in a simple manner. The existing drawbacks of TPSN are: (1) the overhead of creating a tree-based structure to assign the level tags and also broadcasting the two-way messages to synchronize the nodes is relatively high for WSN, (2) adding new sensor nodes to the network might increase energy consumption significantly as TPSN should perform the level-discovery phase to assign new level tags to the new nodes.

Scalable Lightweight Time synchronization Protocol (SLTP) for wireless sensor networks is another synchronization protocol that uses passive clustering and linear regression to synchronize sensor nodes [13]. This protocol has two phases: configuration and synchronization. During the first phase, SLTP utilizes a passive clustering approach to create a set of clusters in the network. At the beginning, a node is selected either dynamically or statically as the reference node to broadcast configuration message with a boolean flag. The initial value of flag is 0 and each node that receives it becomes a cluster member. Then, cluster members set the flag to 1 and re-broadcast it. Each node that receives it becomes a cluster head recursively. In the second phase, the cluster heads broadcast the synchronization packet frequently at specific intervals. When the cluster members receive the packet, they firstly record the local time of transmission and then estimate the clock skew and offset with the sender using linear regression. However, SLTP drawbacks are: (1) it is not very accurate because firstly it just estimates the offsets as well as skews between the nodes and secondly ignores measuring some of the time parameters like access, transmission and propagation time, (2) this protocol is not very suitable for dense and large networks that operate over a long period because the message conflicts between cluster heads as well as members may increase skew and synchronization errors.

Passive Cluster based Clock Synchronization (PCTS) in sensor networks also presents a two-phase protocol to synchronize sensor nodes [14]. First, PCTS divides the network into a set of clusters using a passive clustering approach. Then, the cluster heads collect their cluster members' local time. Afterwards, the cluster heads calculate the average time of all nodes and send it to their cluster members as the reference time. Although PCTS is scalable and simple to set up, it has some drawbacks that might reduce the accuracy and efficiency of synchronization: (1) the overhead of creating and maintaining the clusters is high and may threaten the network lifetime especially for large and dense networks, (2) PCTS does not measure send, receive and reception time for synchronization, (3) sending the inter/intra-cluster messages can increase the network traffic which consequently may increase the reception time. Because of this, and due to the fact that reception time is not included in calculating the synchronized time, skew errors between nodes are increased especially when the network operates over a long period of time.

According to the literature, synchronization traffic overhead is the main drawback of the conventional protocols. This overhead might be due to broadcasting frequent synchronization messages or constructing the hierarchical infrastructure to synchronize the sensor nodes. The drawback may increase the transmission ratio throughout the network that consequently reduces the efficiently of protocols energy. In addition, some of the protocols may ignore considering the skew and set-up errors by estimating the time parameters. Owing to these issues, we are motivated to propose a novel and more efficient time synchronization protocol which is able to synchronize sensor nodes accurately.

3 HRTS Approach

HRTS is a time synchronization protocol that synchronizes the clock of sensor nodes based on their demands. It means that, this protocol allows the sensor nodes be synchronized when they ask for synchronization based on their application or synchronization scenario. HRTS is a three-phase protocol that firstly measures communication delays for the sensor nodes locally (set-up phase). Then, it collects the timing parameters for the sensor nodes that intend to be synchronized based on their demands in a multi-hop manner (updating phase). Finally, HRTS synchronizes the part of network that is defined by the sink for the synchronization (synchronization phase). In this protocol, each node has a unique id, a local clock and two tables to record and monitor the synchronization information: the Information (ITable) and Synchronization Table (STable). These tables (at each node) maintain the required synchronization information about their neighbours. Using the information, each node would be able to set its own local clock with its single-hop neighbours which already are synchronized.

Basically, the ITable provides the information for the nodes to know the messaging delays with each of their single-hop neighbours. It also shows the available multi-hop links between the node and its multi-hop neighbours. HRTS keeps the information in a three-column table: the single-hop neighbours' id of the

node(NID), the end-to-end delay (Edelay) between the node and its single-hop neighbours and finally the ID of multi-hop neighbours (MID). The STable supplies the history information of the synchronization events. This table provides the information for the nodes to avoid transmitting the redundant synchronization messages to the same nodes. The information contains the source/reference nodes' ID (SID), version of synchronization message (V), the list of nodes which are included in the synchronization event and synchronized time (ST).

According to equation 1, Edelay is the messaging delay between two node that has the potential to influence the quality of time synchronization. It means that, the synchronization errors might be increased if Edelay is not measured accurately during the time synchronization procedure. As equation (1) shows, HRTS divides this parameter into two main parts: sender side (SS) and receiver side (RS) times. The parameters are measured locally at sender and receiver nodes. SS includes: send time (S), access time (A), encoding time (E) and transmission time (TT). RS on the other hand involves reception time (RP), propagation time (P), receive time (R), and decoding time (D). We may omit measuring interrupt delays in this paper as it is usually negligible in WSN [15].

$$Edelay = Sender\_delay(SS) + Receiver\_delay(RS) \tag{1}$$

3.1 Set-up Phase

The main objective of this phase is measuring the Edelay between the nodes. This metric would utilized to measure the communication delays during which might influence the accuracy of synchronization procedure. Edelay is calculated at each node locally with its single-hop neighbours. The sensor nodes firstly reset their local timer and then measure their own SS and RS during sending and receiving the AirFrame message respectively. The AirFrame message is the basic and lightweight message frame that is utilized for calculating the local SS and RS at each node. As TT and RP depend on the length of transmission [11], nodes utilize the lightweight AirFrame messages (as reference message) to measure the RS and SS uniformly. It allows them to calculate the delay parameters for the further transmissions by scaling the transmission ratio based on the reference size of AirFrame messages. After calculating SS and RS at a particular time (Si), which is fixed and can be defined by the network's consumer, the nodes measure the Edelay by broadcasting a "hello" message containing the SS, Si and their ID. The receiver nodes calculate the Edelay value by adding its own RS to the received SS for each neighbour. Figure 1 shows how the Edelay is filled by the information which is collected during the setup phase. According to the figure, node 4, for example, updates its own ITable by the Edelay of its neighbours as follow: (node 1, 12ms), (node 13, 1ms) and (node 2, 5ms).

3.2 Updating Phase

This phase discovers the multi-hop neighbourhood around the source node to establish the synchronization region. In other words, the nodes find the available

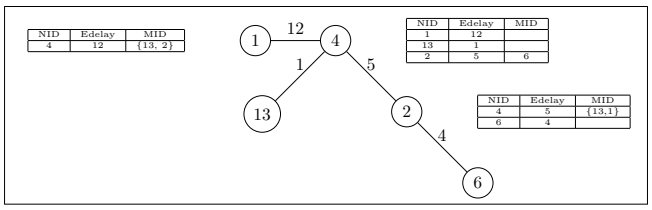


Fig. 1. An example for ITables

communication links to their multi-hop neighbours in their vicinities on demand of synchonization. Based on the collected information, sensor nodes would be able to perform partial synchronization within the particular regions of network that ask for synchronization in the case of adding or replacing new nodes.

After a specific interval (H) (should be greater than all Si because the updating phase should be performed after all setup executions), the nodes which are residing in the synchronization regions broadcast an updating message including their ID as well as NID column of their ITables (their single-hop neighbours). These regions are established by the sensor nodes that ask for synchronization. When the neighbour nodes receive the updating message, they find any row whose first field is the same as the sender ID in its ITable. If the row is found, the node updates its ITables by adding the received NID to its MID column. Figure 1 shows the status of ITables at the end of updating phase. For example, node 4 broadcasts an update message like ({4, {1, 2,13}}), which contains its single-hop neighbours and then the receiver nodes (that can be any of node 1, 2 or13) update their ITables subsequently.

3.3 Synchronization Phase

The objective of this phase is synchronizing the sensor nodes which are in the synchronization region. This phase synchronizes any region of network like a cluster of nodes that intends to be synchronized with the source nodes. The structure of synchronization messages is different. They are composed of five main fields: (1) Source node's ID (SI), (2) Sequence number of synchronization message (Sq), (3) Receiver's ID (RI) (to synchronize a specific part of network), (4) Reference Time (ST) and (5) the SS time (SSi) of the last node in the synchronization event.

According to Algorithm 1, when any node receives the synchronization message, it firstly checks its role as a Target, Gateway or T-Gateway. The node is Target if the sink node puts its address in the message to synchronize. The node is Gateway if it is supposed just to broadcast the message until the Target nodes in its neighbourhood receive. The node is a T-Gateway node if a Target node relays the synchronization message to other ones. These roles define how the nodes should deal with the messages. In the Target case, the receiver should be synchronized with the source node using the information of the message, whereas the receiver just broadcasts the message to inform other multi-hop neighbours

```
input  : Node's ID, ITable and synchronization message{SI, Sq, RI, ST, SSi}
output: Role
if NodeID ∈ RI then
    Role ← Target;
    if (NodeID ∈ NID) or (NodeID ∈ MID) then
    Role ← T-gateway;
end
else Role ← Gateway;
```

Algorithm 1: Node's role check Algorithm

```
input  : Node's STable, ITable, Role and Message {SI, Sq, RI, ST, SSi}
output: updated STable
if SI ∈ STable(SID) then
    if (Message(Sq) > ST(V)) then
        go to the Synchornization Algorithm ;
    end
    else
        the message is old, ignore that;
    end
end
else
    add new row to STable using {SI, Sq, RI, Synchronized Time};
    Go to Synchronization Algorithm;
end
```

Algorithm 2: Freshness check Algorithm

in the Gateway case. If the node is T-Gateway, it should do both tasks respectively. In this case, the node firstly synchronizes itself by the source node and then broadcasts the message to synchronize other ones.

After the role check, the nodes check the freshness of synchronization message using Algorithm 2. If the synchronization message is being sent to the particular group of nodes for the first time, the SI field of message does not match with any SI column of the receiver nodes' STables. So, the receiver nodes record a history of the synchronization message in their STable as a new synchronization event. Otherwise, the message is not new and the receiver nodes just update the current record using the information of the message. At the end, sensor nodes synchronize either themselves or their neighbours with the source node based on their role using Algorithm 3. Target nodes just update their local time and discard the messages, Gateways do not change their local information and just broadcast the message for their neighbours and G-Target nodes broadcast the messages after synchronizing and updating their local time as well as information of the local tables.

4 Discussion and Evaluation

4.1 Simulation

To test HRTS, We utilize the OMNET++ [16] simulator. At first, we setup a network with 32 sensor nodes which are distributed randomly on a 150×150 metre field. According to table 1, the whole simulation time is 1000 seconds for each experiment and the first 50 seconds of each experiment is utilized for the network initialization (network deployment and performing set-up and updating phases). Si (the setup phase initialization time) and H (the updating phase initialization time) are set to 15 and 35 seconds respectively. The initial energy

```
input : Node's STable, ITable, Role and Message {SI, Sq, RI, ST, SSi}
output: Updated STable and Synchronized Network
if (Role − is − Target) then
    { New Time = SSi+Edelay;
    Update the current record in STable with (SI, New Sq, RI, ST, New Time);
    }
end
else if (Role − is − Gateway) then
    {New Time = SSi+Edelay;
    Add ID to RI ;
    SSi←New Time;
    Broadcast the message;
    }
end
else if (Role − is − TGateway) then
    {New Time = SSi+Edelay;
    Add ID to RI;
    SSi←New Time;
    Update the current record in STable with (SI, New Sq, RI, ST, New Time);
    Broadcast the message;
    }
end
```

Algorithm 3: Synchronization Algorithm

Table 1. Experimental setup parameters

Parameters	**Node Number**	**Radio Range**	**Distribution Model**
Values	32 (64, 128, 256, 512, 1024)	20 meters	Random
Parameters	**Sink location**	**Simulation time**	**Iteration**
Values	Center 75X75 (750X750)	1000 s (50 s for initialization)	5 times per each scenario
Parameters	**Initial Energy**	**Field Size**	**Energy/Communication Model**
Values	10 j	150×150 (1500×1500)	Mica2 mote

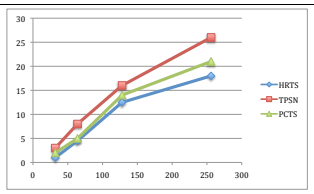
value for each node is 10j based on the Mica2 energy model [17] to transmit and process data. The nodes' radio range is 20 meter and they utilize the Mica2 mote communication model [18] to measure the communication delays. At the end of simulation time, 96 percent of the synchronization messages are delivered correctly and so 31 nodes are synchronized accurately. The unsynchronized node performs set-up and updating phases accurately, but it might miss the synchronization messages due to the message conflicts. The experiments are repeated five times.

In the second step, we increase the size of network to 64,128 and 256 sensor nodes respectively to test the scalability of HRTS. In this experiment, the number of unsynchronized nodes are increased when the size of network is increased. In other words, increasing the connectivity degree (the number of connected nodes) between sensor nodes increases the message conflicts. Figure 2(A) shows the number of synchronized nodes that we got by performing our experiments for five times based on random node distribution scheme. As figure 2(B) shows, HRTS outperforms PCTS and TPSN in the case of synchronization error rate (number of unsynchronized nodes). The synchronization errors are increased when the network becomes denser because the number of delivered synchronization messages is decreased due to the message conflicts.

In the third step, we increase the size of simulation field to evaluate the efficiency of our work in larger networks. In this case, 32, 64, 128, 256, 512 and 1024 nodes are distributed randomly in a 1500×1500 metre field. Then,

Execution No.	32 N	64 N	128 N	256 N
First	31	59	116	244
Second	31	61	111	240
Third	32	58	118	238
Forth	31	60	119	232
Fifth	30	60	117	240

A



B

Fig. 2. A: number of synchronized nodes using HRTS (150×150) in each iteration, B: synchronization errors (X-Axis: The size of network, Y-Axis: the number of unsynchronized nodes).

we measure the synchronization errors similar to the previous experiments. As figure 3 shows, the error is increased when the network is sparse because a set of sensor nodes might miss the messages that the source node sends for either synchronization or infrastructure establishment. It means that a group of nodes, which are residing out of the radio range of source node, cannot be synchronized as they are not able to communicate with the source node to get the reference time. In PCTS, although the nodes might be able to establish the clusters, they cannot be synchronized correctly as the cluster-head might be disconnected from the source node. In TPSN, the unsynchronized nodes cannot receive the level discovery messages to establish the tree-based synchronization infrastructure and so they cannot receive the synchronization messages accordingly. In HRTS, the nodes might miss the synchronization messages although they might perform set-up or updating phases correctly. For this reason, connectivity is one of the most critical issues on the performance of the synchronization protocols. Our protocol works efficiently when the network is fully connected as each node would be able to receive the synchronization messages correctly from its either single-hop or multi-hop neighbourhood. On the other hand, the error might be increased sharply when we add more nodes and the network becomes denser (512 nodes and more). In this case, the density is the reason for increasing the number of errors as the higher traffic overhead increases the message failure ratio. When the connectivity degree is increased between the nodes, they might receive multiple messages at the same time which increases the message conflicts.

4.2 The accuracy

The conventional time synchronization protocols, like RBS, TPSN and SLTP, usually synchronize sensor nodes by estimating skew between the nodes through linear regression methods. In these protocols, the nodes measure their offset with the reference node and then estimate the synchronized time using linear regression methods to normalize the clock drift as well as offset after receiving time reference message. The protocols also may ignore measuring some of the messaging delays. For example, RBS [12] does not consider send or access time

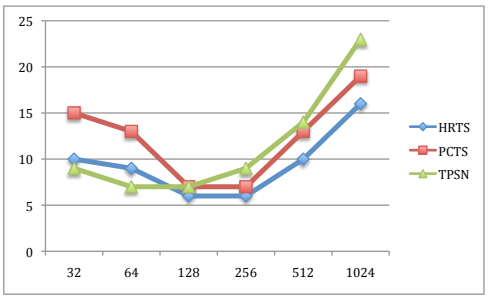


Fig. 3. synchronization errors 1500×1500 (X-Axis: The size of network, Y-Axis: the number of unsynchronized nodes)

for estimating the synchronized time. PCTS [14] ignores measuring reception and receive time parameters and SLTP [13] does not measure transmission and propagation delays. On the other hand, HRTS enhances the accuracy of time synchronization by measuring and involving all these parameters for synchronizing the network's nodes. In HRTS, sensor nodes measure the local time by calculating time skew between the reference and nodes using their clock drift and offset as equation (2) shows. According to the equation, the local clock of node i (C) can be calculated by measuring local clock's drift a(t) and offset b. The clock drift (a) denotes the clock frequency and the clock offset (b) is the difference of node's local clock and the real time.

$$C_i(t) = a_i(t) + b_i \tag{2}$$

Our approach calculates the parameters accurately during the set-up phase based on the properties of the hardware and the length of transmissions (the AirFrame messages). Moreover, HRTS might decrease the synchronization errors caused by the uncertainty of the delays. For example, access time is a key time parameter which depends on the network traffic and has the potential to influence the accuracy of synchronization. Traffic ratio and message queuing delays may increase its uncertainties which consequently increase synchronization errors. For this reason, HRTS reduces the queuing delay by decreasing the network traffic and the number of messages. For example, if we assume N and X as the total number of single and multi-hop neighbours for a particular node like Node A , the node will receive just M (M is less than N and shows the number of the neighbours which send the hello messages at time T) messages during the setup phase in HRTS. On the other hand, the node might receive greater number of messages from either multi-hop or single-hop neighbours (X+N) during the network deployment in TPSN and PCTS.

4.3 Fault tolerance

System fault tolerance is another issue for time synchronization protocols. As drift and skew errors may increase the communication delays between nodes [19] and these delays increase the fault ratio in synchronization procedure especially over long time intervals, proposing a fault tolerant approach to deal with these errors is needed. Frequent updating of the nodes' local clock is a solution to decrease both drift and skew errors and consequently improves the fault tolerance of the synchronization protocol [20, 21]. However, this method might reduce the lifetime of WSNs as these frequent updates need consuming a great deal of sensor nodes' energy to process and communicate. For example, updating the reference time needs re-constructing the tree-based communication infrastructure in TPSN, or exchanging reference messages with all sensor nodes in the network in RBS. So, this solution is not effective to resolve the fault tolerance problem in WSN as it may consume a great deal of energy to process and transmit updating messages through the network.

HRTS enhances the fault tolerance of network because this protocol has the ability to update the time values in a lower cost manner. As the sensor nodes record the information of their neighbours including their IDs and timing information, HRTS would be able to remove any skew or drift error by sending just a new reference time message to update the nodes' clock time. In this case, the problem can be solved when the sensor nodes update their local time based on the new version of time synchronization message. HRTS decreases the energy cost of updates to a considerable degree by reducing the traffic overheads that other protocols might have due to reconstructing the synchronization infrastructure.

4.4 Scalability

HRTS provides a high degree of scalability. As it has the ability to synchronize or update the nodes' clock based on the nodes' demand, new sensor nodes can be synchronized accurately and efficiently beyond either their time or location of placement. It means that when new nodes are added into the network, they just need to establish a connection with any sensor node of the network. Then, the new nodes would perform the first and second phases of HRTS respectively to update their ITables. As HRTS allows the sink node to synchronize a part of network separately and without involving other sensor nodes' clocks, the new nodes can be synchronized easily by receiving the synchronization messages which are broadcasted by the sink. It means that, firstly the sink may put the address of new nodes in the synchroniztion message and then broadcasts it into the network. Then, the nodes that are addressed for synchronization are Target nodes to update their time accordingly using the message. Otherwise, the nodes just broadcast the message like Gateway nodes until the target ones receive. So, our approach presents a low-cost scalable time synchronization protocol which has the ability to synchronize the new sensor nodes efficiently and without consuming a great deal of energy for reconstructing the synchronization infrastructures.

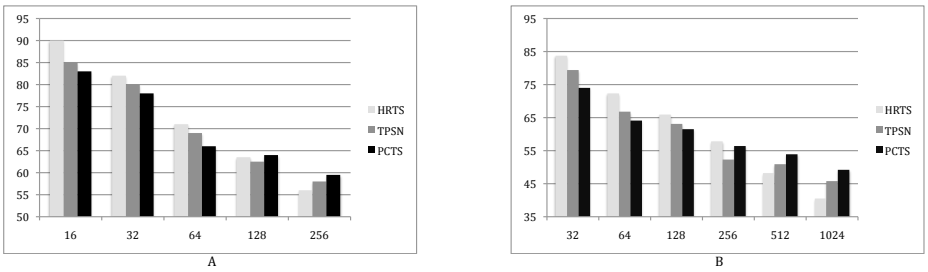


Fig. 4. energy consumption (A: field 150×150, B: field 1500×1500) X-Axis: The size of network, Y-Axis: remaining energy

4.5 Energy Efficiency

Time synchronization protocols aim to conserve the network's energy by minimizing the computation as well as communication overhead of the synchronization [22]. According to figure 4, HRTS is more energy efficient when the network is not very dense. PCTS and TPSN may consume more energy because firstly both the conventional protocols consume a great deal of energy during their network deployment phases to establish either the tree or cluster based synchronization infrastructures. Secondly, the sensor nodes should consume energy to keep the communication paths available during the synchronization procedure.

The network's node number (density) is the main issue that may influence the energy consumption of HRTS. According to figure 4, HRTS outperforms both PCTS and TPSN when the network is not very dense, whereas the protocols are more energy efficient when the network density is very high. HRTS is able to save up to 10 percent of network energy when the node connectivity is not very high because sensor nodes consume less energy for performing the updating phase or network deployment comparing with other ones. It means that the sensor nodes may transmit less number of messages to discover and synchronize their local vicinity comparing with TPSN and PCTS which establish the hierarchical synchronization infrastructure like tree or cluster. On the other hand, the energy efficiency of HRTS is decreased when the connectivity degrees between the nodes is increased. In this case, the nodes should consume more energy to collect the neighbourhood information during their updating phase. Moreover, the intermediate nodes might transmit a set of unnecessary synchronization messages to synchronize the particular nodes through different paths which are established during the updating phase. Although the nodes are able to select the least latency (low cost) path to transmit the synchronization messages and discard the unnecessary messages, the overhead of performing local processing tasks and receiving unnecessary transmissions might increase their energy consumption significantly.

Apart from these issues, HRTS can be considered energy efficient because it has the potential to synchronize just the unsynchronized nodes in the network

and without consuming energy to re-perform the synchronization algorithm on all the network's nodes. Due to frequent topology changes in WSN, it is highly required that the time synchronization protocol would be able to synchronize just the changed group of nodes instead of re-synchronizing the whole of network. TPSN and PCTS protocols may need more energy to update the synchronization infrastructure when the network topology changes. For example, if we add a number of nodes into the network, both the protocols should perform either level discovery or clustering algorithm again to establish the synchronization platform. So, they consume energy to synchronize even the nodes that already have synchronized. On the other hand, HRTS has the ability to synchronize just the subset of nodes which are included in the RI field of updating message rather than involving all nodes. In this case, the network consumes less energy to synchronize the nodes because just the new ones and the sensor nodes on the path (between the sink and the target nodes) consume energy to calculate time parameters, collect the neighbourhood information and transmit the synchronization messages respectively.

5 Conclusion and Future work

Our time synchronization protocol proposes an accurate and efficient approach to synchronize the local time of wireless sensor nodes on demand. This protocol synchronizes sensor nodes accurately by utilizing different time parameters that might are measured during the message transmission procedures. As the protocol has the ability to reactively involve just a part of network's sensor nodes, it has the potential to decrease the communicational and computational overheads and consequently save overall network energy. Table 2 summarises a qualitative assessment of time synchronization protocols, suggesting our approach has the potential to deliver better performance, efficiency and accuracy especially when the network is not very dense. Apart from scalability and fault tolerance, the comparison shows that our approach offers a good fit with WSN due to decreasing skew errors by incorporating more timing parameters over optimal communication links. However, synchronization error propagation is a drawback of HRTS. The number of unsynchronized nodes might be increased hierarchically in the network when the nodes either calculate or receive any false synchronization information (like set-up time errors or wrong information about the synchronization vicinity) during their set-up and updating phases. In this case, the nodes may propagate the synchronization error throughout the network as the receiver nodes calculates their time based on the receiving synchronization messages.

In addition to the points that we mentioned, HRTS can be beneficial in routing, network coverage and clustering as below:

- **Routing**: ITable and STable provide a set of useful information about the sensor nodes neighbourhood that would be influential on routing approaches. In this case, each node has the ability to find its shortest path (less end-to-end delay) with other nodes in either its single or multi-hop neighbourhood.

Table 2. The concluded comparison between our and current time synchronisation protocols

Parameters	RBS	TPSN	PCTS	STLP	HRTS
Scalability	Low	Low	Medium	High	Very High
Fault Tolerance	Low	Low	High	Low	High
Considering Skew errors	No	No	No	No	Yes
Multi-hop synchronization	No	Yes	No	Yes	Yes
Synchronization's traffic	synchronizing nodes	all nodes	cluster members	cluster members	synchronizing nodes in the synchronization session
Energy- efficiency	Low	Low	High	Medium	High
Accuracy	Low	Medium	Medium	High	High

- **Network coverage**: HRTS might be able to control the network coverage by utilizing the information of ITable that shows the connectivity degree between nodes. In this case, we can find the disconnected node, if its first column of ITable is empty. Moreover, we can monitor the density of nodes in the network to control energy consumption by checking the first and third column of ITables.
- **Clustering**: HRTS might be used to find the cluster head nodes as it is able to introduce the nodes which have more stable connectivities. ITable and STable shows the number, quality and availability of these connections.

References

1. M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 413–420, 2006.
2. M. sharifi, S. P. Ardakani, and S. S. Kashi, "Skew: An efficient self key establishment protocol for wireless sensor networks," *International Symposium on Collaborative Technologies and Systems (CTS 2009), Baltimore, Maryland, USA, May 18-22*, pp. 250 – 257, 2009.
3. K. SOHRABY, D. MINOLI, and T. ZNATI, *Wireless Sensor Network, Technology, Protocols, and Applications.* John Wiley & Sons, Inc., Hoboken, New Jersey, 2007.
4. F. G. Khan, M. Corrado, B. Montrucchio, and A. Saeed, "Gossip-based supervision for wireless autonomic networks and services," *6th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, October 18-19*, pp. 376–81, 2010.
5. D. D. Hwang, B.-C. C. Lai, and I. Verbauwhede, "Energy-memory-security tradeoffs in distributed sensor networks," *Third International conference on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW), Vancouver, Canada, July 22-24,*, pp. 70–81, 2004.
6. P. Ranganathan and K. Nygard, "Time synchronization in wireless sensor networks: A survey," *International Journal Of UbiComp (IJU)*, vol. 1 (2), pp. 92–102, 2010.

7. B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.
8. O. Simeone and U. Spagnolini, "Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP Journal on Wireless Communications and Networking, Springer*, vol. 2007(1), p. 57054, 2007.
9. S. M. Lasassmeh and J. M. Conrad, "Time synchronization in wireless sensor networks: A survey," *IEEE SoutheastCon 2010*, pp. 242 – 245, 2010.
10. S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *In Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 138–149.
11. M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 39–49.
12. E. Jeremy, G. Lewis, and E. Deborah, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, December 2002.
13. S. N. Gelyan, A. N. Eghbali, L. Roustapoor, S. A. Y. F. Abadi, and M. Dehghan, "SLTP: scalable lightweight time synchronization protocol for wireless sensor network," in *Proceedings of the 3rd international conference on Mobile ad-hoc and sensor networks*, ser. MSN'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 536–547.
14. M.-O.-R. Md., H. C. Seon, and I. Chi-Hyung, "Passive cluster based clock synchronization in sensor network," in *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 340–345.
15. J. S. Heidemann and R. Govindan, "Embedded sensor networks," in *Handbook of Networked and Embedded Control Systems*, D. Hristu-Varsakelis and W. S. Levine, Eds. Birkhäuser, 2005, pp. 721–739.
16. OMNET++. (Retrieved November, 2012) http://www.omnetpp.org/.
17. D. Scherba and P. Bajcsy, "Communication models for monitoring applications using wireless sensor networks," Automated Learning Group at University of Illinois (Urbana-Champaign), Tech. Rep., 2004, April 02.
18. I.-K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y.-C. Wu, "Clock synchronization in wireless sensor networks: An overview," *Sensors*, vol. 9(1), pp. 56–85, 2009.
19. W. Lan and X. Yang, "A survey of energy-efficient scheduling mechanisms in sensor networks," *Mobile Networks and Applications*, vol. 11, pp. 723–740, October 2006.
20. G. Gaderer, P. Loschmidt, and T. Sauter, "Improving fault tolerance in high-precision clock synchronization," *IEEE Trans. Industrial Informatics*, vol. 6, no. 2, pp. 206–215, 2010.
21. H. Lönn, "A fault tolerant clock synchronization algorithm for systems with low-precision oscillators," in *Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, ser. EDCC-3. London, UK: Springer-Verlag, 1999, pp. 88–105.
22. G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.